

# 繪製函式圖：認識數學函式

last modified September 18, 2011

能夠在畫面上看到繪製的數學函式，可以拉進與數學間的疏離感。如果能夠親自動手操作電腦畫出在書本上曾經看過的函數圖形，那會是件令人興奮的事。本單元試圖讓學生了解電腦繪製函式圖形的原理，並藉此熟練 MATLAB 這個數學軟體的特色，特別是它的矩陣表示法。

## 本章將學到關於程式設計

變數的觀念與設定、MATLAB 的四則運算、變數矩陣的索引、函數的計算與繪圖的觀念。另外在 MATLAB 軟體的使用及繪圖功能也有一些簡單的介紹，方便初學者儘早掌握 MATLAB 所提供的優勢。

〈本章關於 MATLAB 的指令與語法〉

操作元 (operators): + - × / . ^ : ;

指令: zeros, ones, eye, diag, plot, polyval, sin, cos, exp, sqrt, nthroot, pi, set, xlabel, ylabel, title, grid, text, gtext, ezplot

語法: 矩陣的建立

## 1 練習

MATLAB 以描點法來繪圖。所以對二維圖形而言，必須先產生函式上對應的  $(x,y)$  點。然後將這些點以符號的方式顯示在一定範圍的座標軸上。下面的範例將協助你成功的在 MATLAB 畫出一張簡單的圖。

---

範例 1. 先練習如何在 *MATLAB* 的環境下 (命令視窗) 建立矩陣 (含 *scalar*, *vector* and *matrix*)。分別建立定義如下的  $A, B, C, D, E, F, G, H, I, J, K$  11 個矩陣，並觀察結果。輸入方式：

$$A = [1 \ 2; 3 \ 4], \quad B = [1 \ 2 \ 3], \quad C = [5 \ 6 \ 7 \ 8]', \quad D = 1,$$
$$E = C(1:2), \quad F = C(3), \quad G = A(1,2), \quad H = A(:,1), \quad I = A(2,:),$$
$$J = 0:5, \quad K = -5:0.5:5$$

---

注意：

1. 在 MATLAB 的語法中，上述代表矩陣的  $A, B, \dots, K$  稱為變數，其內容 (值) 可以透過等號 (=) 右邊的資料賦予，且隨時可以被改變。請注意等號右邊的資料可以是數值、向量或矩陣資料，或者也可以一律視為矩陣，因不管是數值或是向量都是矩陣的特殊型態。
2. 當變數被賦予內容後，在命令視窗下輸入變數名稱，將可顯示出其內容值。
3. 每個指令的最後若加上分號 (;)，則運算的結果將不會顯示在命令視窗上。
4. 請留意等號 (=) 右邊賦予矩陣資料時的符號 (, : ;)，從執行的結果去瞭解其代表的意涵，並記錄下來。
5. 變數  $E, F, G, H, I$  是取其他變數的部分內容，請特別留意矩陣的索引方式及結果，對於往後 MATLAB 的使用非常關鍵。MATLAB 的矩陣索引與理論上一致，指令  $A(i, j)$  代表第  $i$  (橫) 列，第  $j$  (直) 行的元素，如圖 1 所示。圖中  $A$  代表一個  $5 \times 5$  的矩陣，指令  $A(2, 3)$  代表第 2 列第 3 行的值，也

就是 7。MATLAB 除提供與理論上相同的索引方式，另可採一維的索引法，圖中每個位置右下角的編號，代表該矩陣一維的索引值，所以指令  $A(12)$  與  $A(2,3)$  會得到相同的結果。

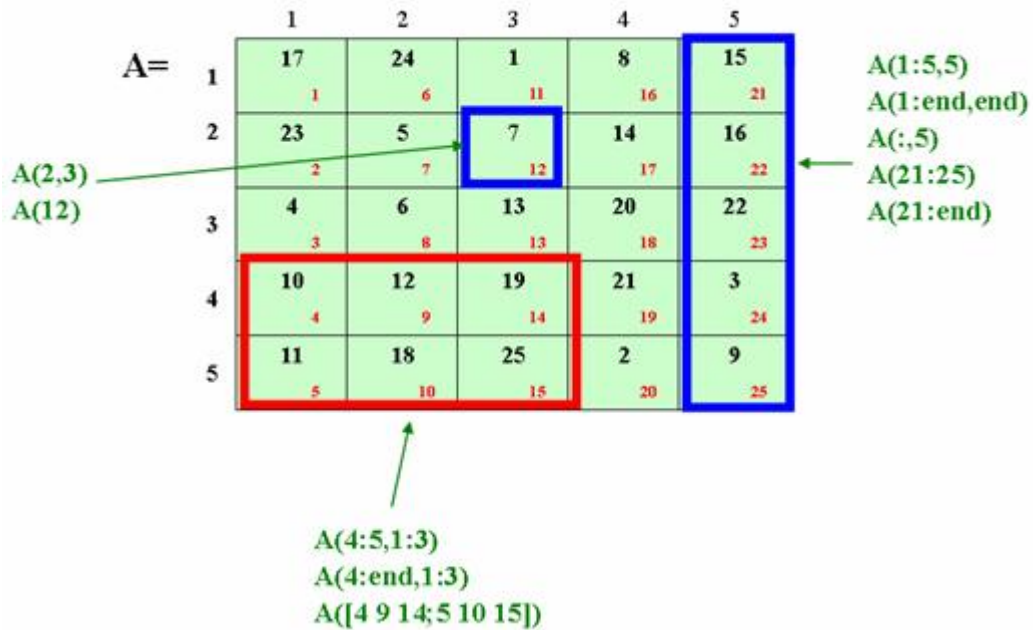


圖 1: MATLAB 的矩陣索引

此外，MATLAB 提供更多的索引方式，方便擷取更多的資料，圖 1 也顯示擷取第 5 行的 5 種方式，其中保留字 'end' 代表該行或該列的最後一個索引位置，方便下指令時不需要先知道矩陣的長度。圖中也展示如何擷取一個子矩陣。不管是從矩陣中擷取一個值、一個向量或是一個子矩陣，初次接觸的使用者，不妨實際在命令視窗上逐一輸入，並觀察結果，對照指令，相信一定可以心領神會，比起在此說得九彎十八拐要來得有用。練習時，可以指令  $A = magic(5)$  來產生圖中的矩陣。

6. 利用 MATLAB 的指令 whos 觀察每個矩陣的大小。
7. MATLAB 也提供一些指令可以更迅速的產生特殊的矩陣，試試看這些指令:zeros, ones, eye, diag。使用前可以利用 help 的指令了解使用的方式，

譬如 help zeros。

---

範例 2. Let  $A = [1 \ 2; 3 \ 4]$ ,  $B = [5 \ 6; 7 \ 8]$ ,  $c = 3$ , 逐一執行以下的運算練習並觀察結果。

$A + B$ ,  $A - B$ ,  $A + c$ ,  $A * B$ ,  $A/B$ ,  $A^c$ ,  $c * A$ ,  $A/c$ ,  
 $A .* B$ ,  $A ./ B$ ,  $A.^c$ ,  $A'$ ,  $B'$ ,  $(A * B)'$ ,  $B' * A'$

---

請注意 MATLAB 所使用的運算元符號, 如  $+ - * / ^ .'$  等所代表的意涵。從執行的結果去觀察並記錄下來。

MATLAB 的數學四則運算與冪方大致符合學理上的矩陣運算, 但 MATLAB 也巧妙的利用矩陣式的資料結構, 設計了一個特殊的運算元 (operator)  $.*$ , 讓多筆資料的運算可以一次完成, 譬如變數  $x, y$  各有 5 個樣本, 其值如下

$$x = [x_1 \ x_2 \ x_3 \ x_4 \ x_5], \quad y = [y_1 \ y_2 \ y_3 \ y_4 \ y_5]$$

要計算兩變數的乘積  $xy$  的 5 樣本值, MATLAB 的指令為  $x .* y$ , 即

$$\underbrace{(x_1 \ x_2 \ x_3 \ x_4 \ x_5)}_x .* \underbrace{(y_1 \ y_2 \ y_3 \ y_4 \ y_5)}_y = \underbrace{(x_1y_1 \ x_2y_2 \ x_3y_3 \ x_4y_4 \ x_5y_5)}_{x.*y}$$

一個指令同時計算好所有的樣本資料, 這有別一般正規的向量乘積, 必須注意前後向量的大小才能相乘。而 MATLAB 這種元素對元素的乘法 (element by element), 提供大量資料運算時的方便。再舉矩陣的運算為例來區分  $*$  與  $.*$  的差別:

$$\underbrace{\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}}_A * \underbrace{\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}}_B = \underbrace{\begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}}_{A*B}$$

$$\underbrace{\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}}_A .* \underbrace{\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}}_B = \underbrace{\begin{pmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{pmatrix}}_{A.*B}$$

除乘法外，除法 (\) 與乘冪 (^) 都可以這樣使用，不管是向量或矩陣，只要使用的時機恰當，都可以達到很好的效果。以下的內容及單元將陸續透過實際的問題介紹這個特殊運算元的好處。

範例 3. 函式  $y = f(x) = 2x + 1$

1. 當  $x = 0, 1, 2, \dots, 10$ ，分別計算其相對的函數值  $y$ 。
2. 繪製函式圖，其中  $0 \leq x \leq 10$

利用簡單的四則運算，函數值很容易計算出來，如圖 2。

```

Command Window
>> x=0;
>> y=2*x+1

y =

     1
  
```

圖 2: 函數運算

當重複執行這兩個指令並改變  $x$  值,<sup>1</sup> 便可以輕鬆的計算出所有的函數值。不過每次只算一個函數值絕不是強力軟體的作為, MATLAB 利用了矩陣的運算提供方便的計算方式, 如圖 3 所示

```
Command Window
>> x=0:10

x =

     0     1     2     3     4     5     6     7     8     9    10

>> y=2*x+1

y =

     1     3     5     7     9    11    13    15    17    19    21
```

圖 3: 以矩陣運算計算多個函數值。

從圖 3 中  $x, y$  向量很清楚的呈現出其間的函數關係, 這些向量的對應數值形成繪圖時的座標點, 以下說明繪製函數圖形的步驟:

1. 先依繪圖範圍產生  $x$  方向的所有點, 譬如  $x = 0 : 10$
2. 再根據函式的關係計算所有相對應  $y$  方向的座標值, 譬如:  $y = 2 * x + 1$
3. 再利用指令 `plot` 繪製不同型態的圖形, 譬如 `plot(x, y)` 將所有的座標點以直線連接。請特別注意 `plot` 提供的各種繪圖的選項 (利用 `help plot` 或 `doc plot`), 試著去改變描繪座標點的方式、點的符號、顏色等選項。

---

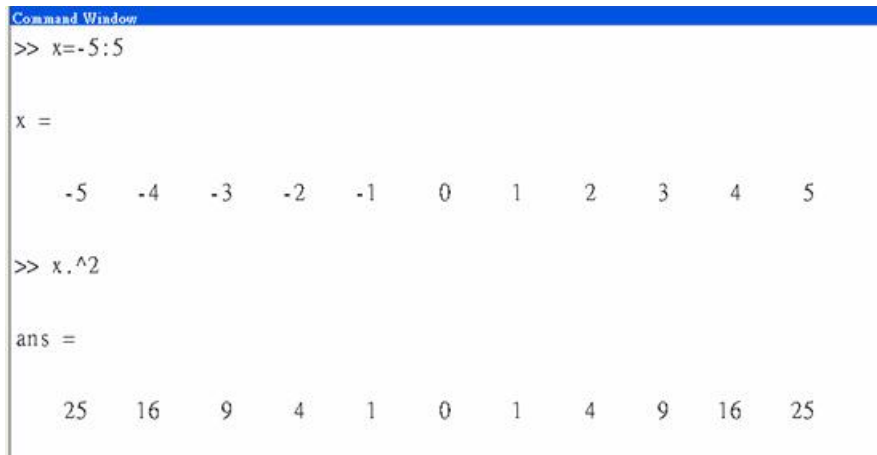
#### 範例 4. 繪製函式

<sup>1</sup>Tips: 由於必須不斷的帶入不同的  $x$  值, 相同的指令必須不斷的被執行, 如果要重複的輸入指令, 將影響實驗的興致。所幸在 MATLAB 的命令視窗中可以利用鍵盤中的  $\uparrow$  鍵, 不斷的將之前執行過的指令秀出來, 方便實驗的進行。例如要重複圖 2 的指令, 可以按兩次  $\uparrow$  鍵回到  $x = 0$ , 將 0 修改為 1, 按「Enter」執行該指令, 隨後再按兩次  $\uparrow$  鍵回到  $y = 2 * x + 1$ , 直接按「Enter」執行便可以得到答案。另外也可以先輸入欲執行指令的前幾個字, 再按  $\uparrow$ , 即可快速找到該指令。

$$1. y = f(x) = x^2 + 3x + 5 \quad -5 \leq x \leq 5$$

$$2. y = f(x) = x^3 - 10x^2 + 29x - 20 \quad 0 \leq x \leq 7$$

前一個範例說明繪製函數圖形的三個步驟，缺一不可，也各有「訣竅」，否則不是畫錯就是畫得不好看，展現不出「一張圖值千字」的價值。對初學者而言，應先學會利用 MATLAB 的特點來計算函數值，特別是 MATLAB 特殊的運算元“.”。例如計算第一個函數值時，可以先試試  $x^2$  的計算方式，待結果正確了才加入其他較簡單的項次  $(3x + 5)$ ，圖 4 展示平方項的計算。



```
Command Window
>> x=-5:5
x =
    -5    -4    -3    -2    -1     0     1     2     3     4     5
>> x.^2
ans =
    25    16     9     4     1     0     1     4     9    16    25
```

圖 4: 函數中計算平方項次的方法。

繪圖的方式同前一題，不過須注意

1. 描繪的座標點「數」；前一個範例的函數是一條直線，當 plot 指令將座標點與點間以直線連接時，看起來毫無問題，但是當函數為曲線圖形時，座標點的距離如果不够近，兩點間的直線連接將會造成圖形的鋸齒狀，不符合函數的特質，因此需要利用更密集的座標點來產生平滑的視覺效果。這需要從 x 軸資料的產生做起，譬如  $x = 0 : 0.1 : 7$ ，從 0 到 7 每隔 0.1 產生一個點，而點與點間 0.1 的間距是否可以造成平滑的曲線端視視覺感受而定，過小並無意義，徒增資料量與計算時間。圖 5 展示不同間距的視覺效果。

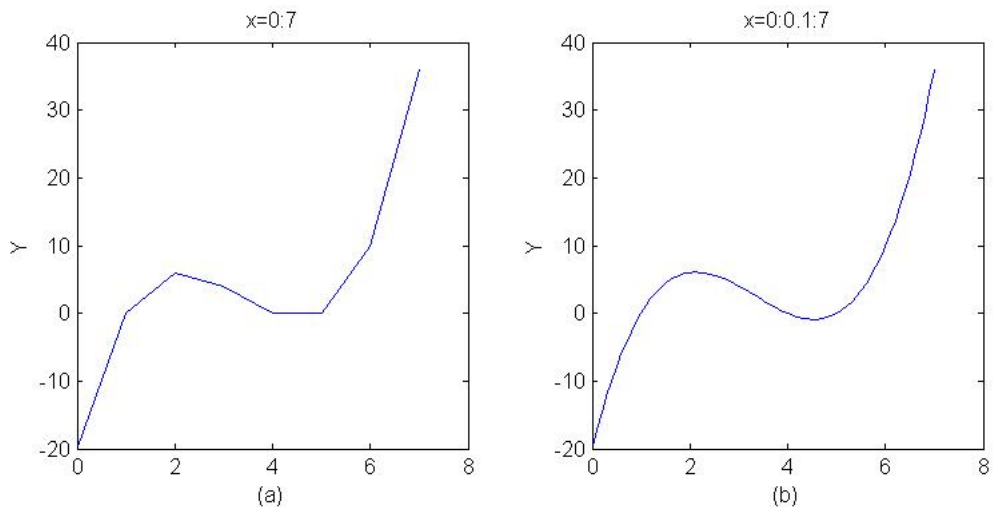


圖 5: 座標點の間距

2. 指令 `plot(x, y)` 中, 變數 `x` 與 `y` 分別代表函數的 X,Y 座標點, 其值的產生也不同。變數 `x` 的內容依據繪圖的範圍與座標點の間距設定一組向量, 如前述的 `x = 0 : 0.1 : 7`, 而變數 `y` 的內容則依函數的關係從變數 `x` 計算得來, 本題的函數具平方與立方項, 可以利用前一個範例陳述的運算元 `.*` 來做乘冪, 一次計算所有 `x` 座標的對應函數值 `y`。譬如指令: `y = x.^2 + 3 * x + 5`, 在此 `x.^2` 的意義請參考範例1、2的矩陣運算。如果只是單純的使用 `x^2`, MATLAB 將出現錯誤的訊息, 告知

```
??? Error using => mpower
Matrix must be square.
```

圖 6: MATLAB 指令執行的錯誤訊息

表示只有方陣 (含數值) 才能做平方, 一般向量的平方是不合法的。

3. 當函數是多項式時, MATLAB 提供一個簡便的指令 `polyval`, 方便計算多項式的函數值, 下列的指令可以取代之前的函數計算:



```
x=-5:0.1:5;
p=[1 3 5];          多項式函數的係數
y=polyval(p,x);
```

關於 polyval 的詳細使用方式，請參考線上使用手冊的說明。

為方便指令的下達與圖形變動的觀察，MATLAB 在 7.x 版以後可以將圖形視窗也整合進來，如圖 7 所示。圖中的視窗切分成四個子視窗，每個視窗都可以點選其右上角的「箭頭」圖示，成為獨立的視窗。子視窗的位置與配置也可以透過滑鼠「拖曳」互相交換或整併。讀者可以試著擺設所有子視窗到自己最習慣觀察的位置。另外，子視窗也可以含多個不同功能的視窗，譬如左上角的子視窗含「Current Directory」與「Workspace」兩個功能視窗。右上角的子視窗含「Array Editor」與「Figures」兩個視窗。透過滑鼠點選標頭即可選取。

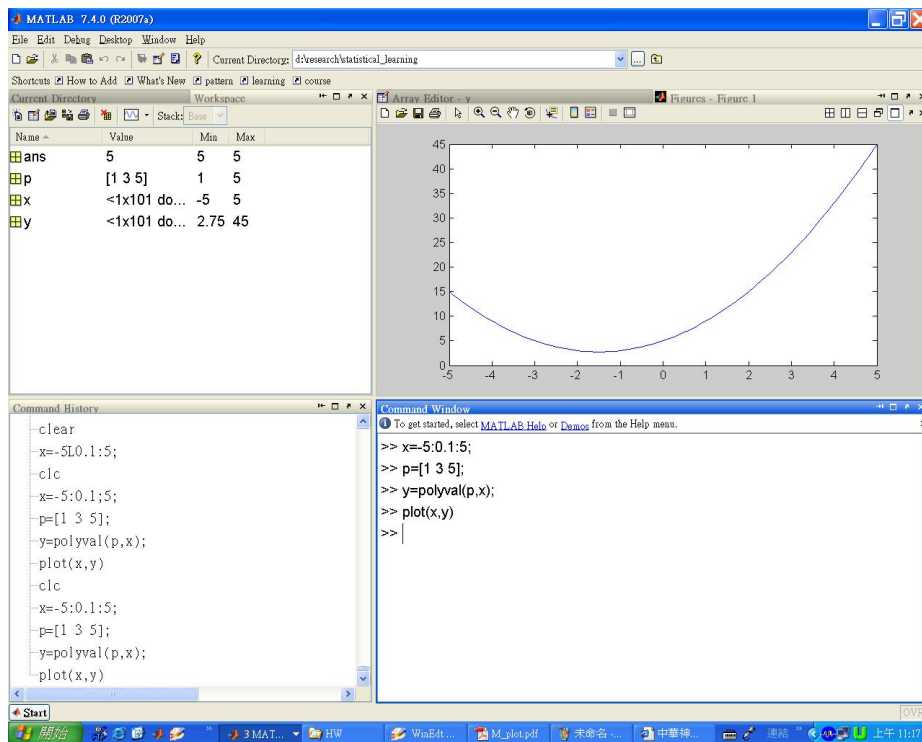


圖 7: MATLAB 整合視窗

範例 5. 繪製函式  $y = f(x) = x^4 - 8x^3 + 16x^2 - 2x + 8$   
請選擇適當的範圍，務必看到比較完整的圖。

---

函數圖的繪製範圍的選取非常重要，太寬、太窄、偏左或偏右都可能看不到該函數的全貌或特質。譬如畫出像圖 9 的函數圖，便因為範圍取得太寬，以致於看不到該函數「最有價值」的最小值部分，因此繪圖時必須不斷的調整範圍，以突顯出函數最值得觀察的部分。

---

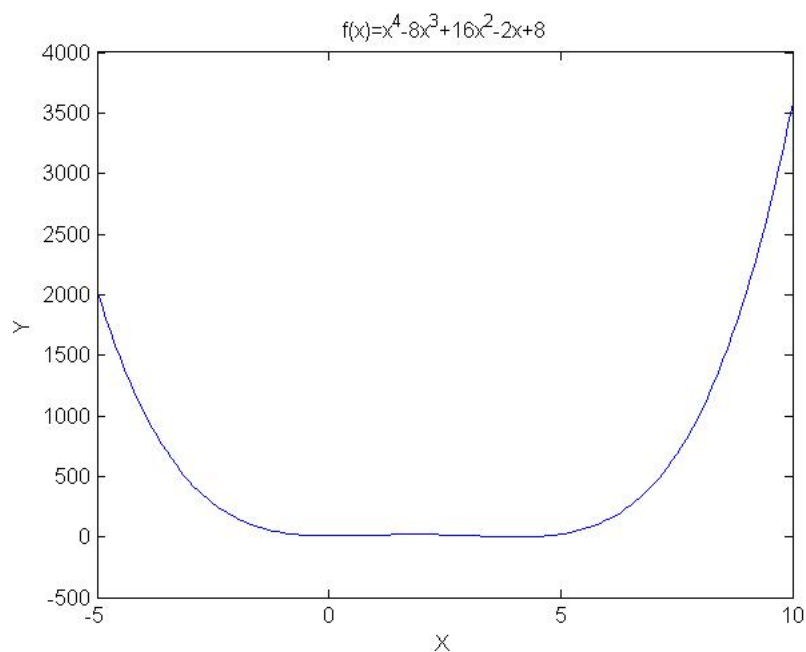


圖 8: 函數圖範圍的決定

範例 6. 依下列範圍繪製函式  $f(x) = x + \sin(x)$

(1)  $1 \leq x \leq 1000$  (2)  $1 \leq x \leq 500$  (3)  $1 \leq x \leq 100$  (4)  $1 \leq x \leq 10$

---

這個範例也是說明範圍選擇的重要，透過不斷的縮小範圍，才能看出函數的特性。

## 2 觀察

1. 前面的範例僅提及如何將資料放入一個變數中, 或是如何從一個資料變數中選取部分範圍。由時候應用上卻需要將資料變數的內容變小, 譬如, 剔除一向量的某一項或某幾項資料, 或是剔除一矩陣的某一行 (或列) 資料, 示範如下

```
a=[1 2 3 4 5];
a(3)=[ ];
此時向量 a=[1 2 4 5];
a([1 3])=[ ];
此時向量 a=[2 5];
*****

A=[1 2 3;4 5 6;7 8 9];
A(2,:)= [ ];
此時矩陣 A=  $\begin{bmatrix} 1 & 2 & 3 \\ 7 & 8 & 9 \end{bmatrix}$ 
```

2. 電腦繪圖係以描點的方式畫上去, 因此點與點間的距離越近, 所呈現出來的圖形越平滑。請觀察在一定的範圍內, 點的數量多與少的效果, 也就是調整  $x$  的間距。
3. 利用 MATLAB 指令計算所畫上去的點數 (即觀察向量變數  $x$  的長度)。譬如 `length`, `size` 等指令。
4. 改變 `plot` 指令的參數, 讓畫上去的點做不同的表現, 譬如, 畫出散佈圖。利用 `help plot` 指令, 看看有那些繪圖的技巧可以使用。
5.  $x$  軸的範圍取捨非常重要, 除了給定的範圍外, 請試試看其他範圍。寬一點或窄一點。

6. 為這些圖加上些裝飾, 如  $x$ ,  $y$  軸的文字、標題或是格線 ( `xlabel`, `ylabel`, `title`, `grid`)。例如圖 9 展示 `title` 的使用, 其指令如下

```
title('f(x) = x^4 - 8x^3 + 16x^2 - 2x + 8')
```

請注意 `title` 所顯示的是文字而非函數的計算, 因此與 MATLAB 的運算元無關, 純粹是文字的編輯, 譬如符號  $\wedge$  代表文字的上標。相關的用法可以查循線上使用手冊。

7. 對一個陌生的函數繪圖, 常要花很長的時間才摸索到適當的範圍, 這時可以使用 MATLAB 指令 `ezplot`, 迅速的得到圖形。做法如下

```
ezplot('x^4 - 8 * x^3 + 16 * x^2 - 2 * x + 8')
```

注意, 引號內的函數輸入方式不同於前面的陳述, 不需要將  $x$  當作向量看待。`ezplot` 雖然迅速方便, 但畢竟是「電腦選的」範圍, 有時候還是看不到函數的細節。不過不失為一個首先嘗試的指令。`ezplot` 內放置函數的方式也可以採用內建函數的方式, 譬如上述的指令可以改為

```
ezplot('polyval([1 - 8 16 - 2 8],x)')
```

MATLAB 提供許多常用函數 (如機率密度函數) 的指令, 用 `ezplot` 來繪製相當方便。

8. 練習將結果 (圖) 貼到 WORD 或其他文書編輯工具 (如 Latex 或中文的 `cwTex`) 裡面, 做好如何呈現結果的工作。將 MATLAB 產生的圖形貼到其他地方的方式有兩種; (1) 利用 `File/Export` (7.0版以前) 或 `Save As` (7.0版以後) 的方式將圖形以檔案的方式儲存下來。(2) 利用 `Edit/Copy Figure` 的方式將圖形 `copy` 下來, 再到 WORD 或其他類似的軟體, 直接以 `paste` 的方式貼回。

### 3 作業

畫出下列函式,  $x$  範圍自訂, 盡量畫出比較完整的函式圖, 並加上必要的裝飾文字。未曾使用過的指令, 請自行利用 `help` 或 `doc` 該指令的方式察看其使用說明。

1.  $y = f(x) = \sin(x) + \cos(x)$ ,      代表 `sin, cos` 的指令分別是 `sin, cos`。

2.  $y = f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$       代表指數的指令 `exp`

3.  $y = f(x) = \sqrt[3]{\frac{4 - x^3}{1 + x^2}}$

開根號的方式有兩種, 譬如計算 $\sqrt{9}$ 的指令可以是:(1) 以幕方的方式 `9^0.5`, (2) 採開平方根的指令 `sqrt(9)`, (3) 採開實數根的指令 `nthroot(9, 2)`。<sup>2</sup>

第1與第3個方式均適用於本題函數, 當使用幕方時須特別注意 MATLAB 幕方指令的下達方式, 譬如指令 `-8^(1/3)` 及 `(-8)^(1/3)` 會產生不同的結果, 何者才是正確的, 從結果很容易辨識, 但卻與一般常理的認知相左, 因此建議除平方根外, 還是採 `nthroot` 指令比較安全, 否則一旦錯誤將很難發現。

4.  $y = f(x) = \frac{1}{x - 1}$

5.  $y = f(x) = \frac{1}{2\sqrt{2\pi}} e^{-\frac{(x-1)^2}{8}}$        $\pi$  的指令為 `pi`

6.  $y = f(x) = x^{2/3} = \sqrt[3]{x^2}$

7.  $y = f(x) = 2x^3 - x^4$

8.  $y = f(x) = x\sqrt{4 - x^2}$

9.  $y = f(x) = \frac{\ln x}{x^3}$

10.  $y = f(x) = 3, 1 \leq x \leq 5$

11.  $x^2 + y^2 = 1$

---

<sup>2</sup>這個指令在7.x 版之後才有。

補充: 下面的指令可以改善圖形呈現的品質

```
set(gca,'xtick',0:1:5)
```

這個指令可以將 x 軸在 0 到 5 之間的刻度間隔改為 1。當然將 'xtick' 改為 'ytick' 就可以針對 Y。

如果想將線條變粗, 使用

```
plot(x,y,'LineWidth',2)
```

裡面的 2 代表線條得寬度, 數字越大越寬。如果還想配上顏色, 可以這樣做

```
plot(x,y,'LineWidth',2,'color','red')
```

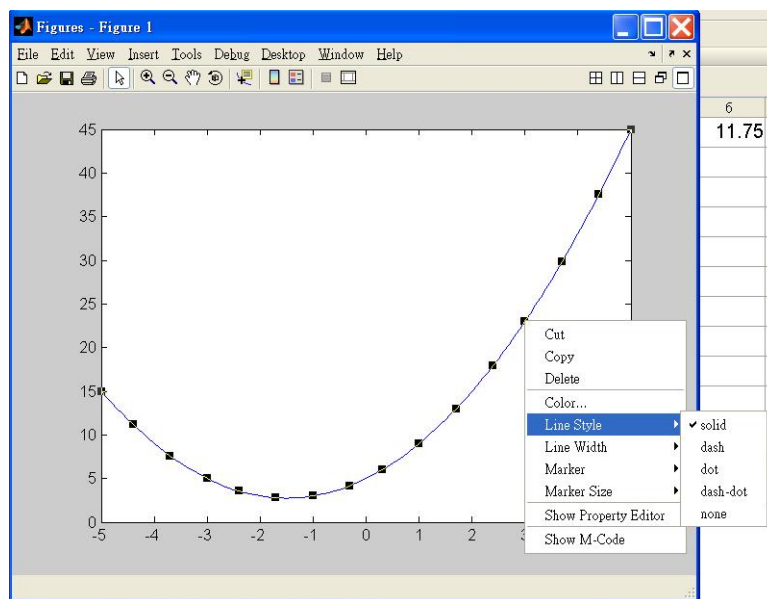


圖 9: 圖形編輯器

MATLAB的繪圖指令非常豐富, 變化很多, 可以從線上使用手冊查詢到。不過對初學者而言, 實無須浪費過多時間在這方面, 看到一個學一個, 夠用就好, 不急於一時, 隨著時間的累積, 將會非常可觀。另一個簡便的方式是直接在繪圖區作圖形的編輯, 如圖 9, 步驟如下:

1. 點選功能選單「Tools → Edit Plot」或直接以滑鼠選取左上方的「箭頭」工具圖示。
2. 選取欲編輯的物件 (線條、符號、或是區域,) 按滑鼠右鍵或雙擊左鍵拉出編輯盤。
3. 選取編輯項目, 進行修改。

利用圖形編輯器的好處是, 不需要知道指令, 憑著編輯盤的選項即可隨意改變圖形。但缺點是動作多, 速度慢, 僅適合一次或不常用的編輯需求。有一個折衷的辦法既可以符合編輯上的需求, 也可以趕上指令編輯的速度。在圖形視窗的選單上找到「File → Generate M-File」, 要求 MATLAB 將目前的圖形轉換成指令。圖 10 中橢圓形圈出的部分即畫出圖中線條的指令。複製這個指令就可以立即產生相同的線條。

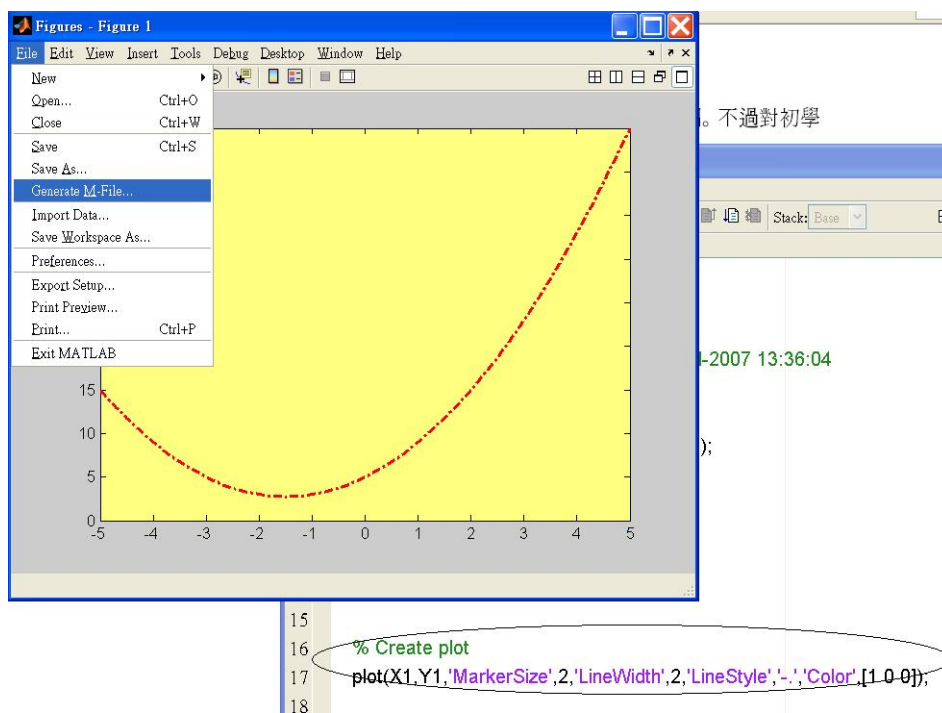


圖 10: 圖形編輯器產生指令的功能

另外, 如果想在 MATLAB 所做的圖上放上比較複雜的數學式, 可以套用 L<sup>A</sup>T<sub>E</sub>X 的指令, 作法如圖 11 所示, 結果如圖 12。圖 11 的 text 指令中的「...」用在指令

太長時作為斷句使用，兩個「\$」號中間放入一般  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  的數式指令。令外一個比較輕便的做法如下，其中「...」的部分填入  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  的數式指令。

```
gtext('$ ... $','interpreter','latex')  
  
>>  
>> text('Interpreter','latex',...  
      'String','\frac{1}{\sqrt{2\pi}}e^{\frac{-x^2}{2}}$',...  
      'Position',[2 .35],...  
      'FontSize',12)
```

圖 11:  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  指令的連結。

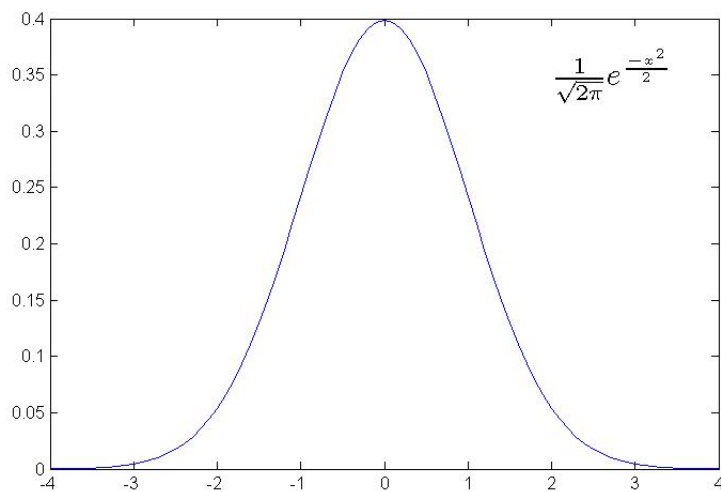


圖 12:  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  指令連結範例。